UNITED STATES DISTRICT COURT

FOR THE WESTERN DISTRICT OF WISCONSIN

* *

WISCONSIN ALUMNI RESEARCH FOUNDATION,

Plaintiff,

-vs-

Case No. 08-C-78

INTEL CORPORATION,

Madison, Wisconsin
August 8, 2008
9:00 a.m.

Defendant.

* *

STENOGRAPHIC TRANSCRIPT OF CLAIMS CONSTRUCTION HEARING
HELD BEFORE CHIEF JUDGE BARBARA B. CRABB

APPEARANCES:

For the Plaintiff: Heller Erhman
BY: ATTORNEY MICHELLE UMBERGER
1 East Main Street
Madison, Wisconsin 53703
ATTORNEYS ANAPAM SHARMA and ROBERT HASLAM
275 Middlefield Road
Menlo Park, California 94025

For the Defendant: Wilmer Hale
BY: ATTORNEYS WILLIAM LEE, STEPHEN MULLER and DONALD STEINBERG
60 State Street
Boston, Massachusetts 02109

Boardman, Suhr, Curry & Field
ATTORNEY RICHARD BOLTON
1 S. Pinckney Street, 4th floor
Madison, Wisconsin 53703

Lynette Swenson
Federal Court Reporter
United States District Court
120 N. Henry St., Room 520
Madison, WI 53703
(608) 255-3821

(Call to order)

THE CLERK: Case Number 08-CV-78. Wisconsin Alumni Research Foundation versus Intel Corporation called for a claims construction hearing. May we have the appearances, please.

MS. UMBERGER: Good morning, Your Honor. Michelle Umberger of Heller Erhman. And with me are my colleagues Robert Haslam and Anupam Sharma. Also at counsel table is WARF's expert, William Dally. And we also have in the courtroom representatives of WARF, Carl Gulbrandsen and Michael Falk. And also one of the inventors on the patent-in-suit, Dr. Gury Sohi.

THE COURT: Thank you.

MR. LEE: Good afternoon, Your Honor. My name is Bill Lee of Wilmer Hale. With me are my colleagues Donald Steinberg and Steve Muller. At the table is Eric Braun, from Fulcrum Legal Graphics, who hopefully will make everything work on the screens for you. With us today are Professor Douglas Clark from Princeton University, who is one of the declarants that Your Honor has, and our local counsel, Mr. Rich Bolton, from the Boardman firm.

THE COURT: Thank you. Okay. I want to start off by thanking you for your briefs. These were two sets of the best briefs that I have received in a long

time and I have to tell you what a pleasure it is to have good briefs.

Second, I think I understand that you want to give me some background on the invention, which would be very helpful, but then I really want to focus on what I think are the two crucial questions or at least the ones that are -- that I've been struggling with. One is this whole question about deciding whether the patent operates solely by using the load/store pairs to identify misspeculations or -- and that's what I understand to be defendant's position -- or whether as plaintiff argues it incorporates the three-tier approach that permits the data speculation circuit to detect data dependence in either individual instructions or instruction pairs.

And the second is the term "in fact executed" and exactly what it means. Does it refer to accessing a memory address? Does it require completion of a process, or may it include an instruction that has been executed as far as possible and is considered ready to commit the operation? Or would it also include certain to access?

So those are the two. I think those two really pervade all of the claim terms that are in dispute, and the other aspects don't seem as troublesome to me. But

perhaps I was missing something.

All right. How do you wish to proceed? Mr. Haslam, do you want to start out?

MR. HASLAM: We had, I think, anticipated that at least the two issues that you had would be the primary issues and we have prepared -- Professor Dally had prepared a short presentation on the background of the technology, but also just walking through how the patent operates because I think it's particularly relevant to the two issues that the Court has asked its questions about. And then I would be prepared to follow-up with a brief argument on the first issue, which is the data speculation circuit and whether it operates only on load/store.

THE COURT: And we will go until twelve o'clock if we need to and we'll split the time. I'll keep track. All right. You may proceed. Before I do that, Mr. Braun or whoever is going to be --

MR. LEE: Your Honor, I'll be doing the argument. I think what we -- perhaps we could do this: Mr. Steinberg, who is one of my partners, is also a computer scientist from Princeton who is going to address some of the same issues as Professor Dally. It might make sense, if it's all right with the Court and Mr. Haslam, to have each of them cover the background

technology, then move to the two terms in dispute so we don't have unnecessary duplication.

MR. HASLAM: That probably is the best way, then the Court has all the background technology.

THE COURT: That's fine. Okay. And you're Mr. Lee, right?

MR. LEE: Yes, Your Honor.

THE COURT: Okay. Mr. Braun was probably terrified when I --

MR. LEE: I was happy to sit here, but --

THE COURT: All right. Mr. Haslam, you may proceed.

MR. HASLAM: Professor Dally, do you want to speak?

MR. DALLY: Your Honor, I'd to start with just a brief description of what the key issues are, and they have to do with data speculation in the high performance processor. Data speculation is where you execute a load instruction out of order with the store instruction before you know whether that load depends on the result of the store or not. This ambiguous dependence occurs, as in the case here taken from Table 1 of the patent-in-suit, where I have a store instruction, in this case the store is the result of some computation to the memory location indicated by the address in register

one, and I have later in program order a load instruction that loads from the address in register two, but I may not yet know those addresses. So the dependence is ambiguous because depending on the contents of register one or register two, the load may depend on the store or it may not.

The conservative thing is to execute in program order, executing a store first and then the load. But if they don't actually depend on one another, it's advantageous for performance reasons to execute the load first.

So here is the conservative approach where the program order is shown on the left and then I show that if I just execute the store first and then the two loads, even if the registers have the same address, I'll get correct execution. But I may wish in some cases to speculate, where I'll execute the load first, in this case I'll execute instruction two first. Even though it may depend on the store because I'm speculating, that is I'm guessing that register two does not have the same value as register one, and this will be a safe out-of-order execution. Or in this case I may speculate on both instructions two and instructions three, executing them both ahead of the store, speculating that both of the load addresses will not match the store

address.

Now because I have to guarantee to the programmer that the machine will execute the program as they intended, if I do that speculation, I have to check, and if I find that one of these addresses matches, that's a misspeculation and I need to then cancel the load that misspeculated and go back and run it again.

THE COURT: Could you say that over again? If you find there's a match --

MR. DALLY: Okay. On the right side here I'm speculating. Let's take the simpler case where I'm speculating on just one load, instruction two, and I'm executing it ahead of store instruction one, that's a safe thing to do as long as the contents of register two does not match the contents of register one. But if I go ahead and I speculate, I execute the load early and I later find that register two contains the same value as register one, then that was an incorrect speculation. There was a dependence, but I executed the load early, reading a stale value from that location, and therefore that's a misspeculation and I have to cancel the load or rerun it.

The patent is all about predicting when that misspeculation is likely so that I can go back so I can do the aggressive approach, the data speculation

approach, which gives me good performance when it's not likely that those two will match. But when it is likely that those two will match, I'll do the conservative approach, executing them in order so I won't have to incur the overhead of canceling and rerunning the load instruction.

The invention builds on prior art. As indicated here, the data speculation circuit is known in the prior art and they've extended the data speculation circuit by adding a set of signals to it that communicate with the predictor that predicts when it's likely or not that misspeculation will occur. And so because they refer to the prior art for the data speculation circuit, one familiar with the art would understand what they meant is what everybody did at that point in time which is they would use load buffers and store buffers to track the load and store instructions that are in what's called the instruction window. Instructions enter the instruction window as they are fetched and as they are committed or retired, they exit the instruction window. And in between, the load buffer and the store buffer track all those in stores that are in some form of process.

THE COURT: And how do you define a buffer?

MR. DALLY: A buffer is a table, a storage

device where I can record these load and store instructions and some of their attributes, and particularly when it's known the addresses that they're reading to or writing from so that I can compare them.

So in this example, after the store instruction would first be fetched, entered into this table, but it's not yet known what address it's storing to, so that location is left blank. Then because we fetch in program order, we fetch instruction two, load it in the table, fetch instruction three, load it in the table, and this shows a point in time where the two load addresses, the value of our two and our three are known, where instruction three has completed, instruction two has been issued to the memory system but is not yet complete and then the store is about to commit, it's determined what its address is, and as is disclosed in the patent-in-suit, when the store commits, it checks its address against that of all concurrent loads, which includes loads that have completed as well as loads that have issued but not yet completed. And so the address, in this case 100, which is what the value of register one turned out to be, is compared against the address that's recorded for the concurrent load operations, and in this case, because there's a match between the store address and the address of the load of instruction two,

instruction two will be canceled or squashed and will have to be rerun even though it hasn't completed. And instruction three does not match, and so it's going to be allowed to commit when it gets to be its turn, since instructions commit in order.

THE COURT: Can you give me a real life example of what you're talking about?

MR. DALLY: So this is pretty close to real life. Here it's basically if I execute a load early, I've executed it ahead of the store, to track, to detect if I did that correctly or not. I have to record in this load buffer, in this table, that this load which occurred at a certain point in program order had a certain address. I will then, if there's a store that's earlier in the program order, when that store commits, that is when it gets to be the oldest instruction in the instruction window and it's time to check if it's safe for that store to commit and write its results to memory, I will check the address of the store against the address of that load and any other load which is later in program order but has already started execution. Because if any of those addresses match, it means that the load misspeculated, that it went early when it was not allowed to go early.

THE COURT: I was just thinking from the point

of view of the person hitting the keys.

MR. DALLY: Oh, okay.

THE COURT: What would happen?

MR. DALLY: Well, the person hitting the keys will never see this.

THE COURT: Oh, I know that. And I'm grateful.

MR. DALLY: The reason is the whole art of high performance processor design is to make the processor do lots of things in parallel, all going on at the same time, and to have them all out of order, but to make it so that if you stop the machine at any point in time, it looks like everything went on one at a time and in order. And therefore even from the point of view of the assembly language programmer, this isn't seen. The instructions and semantics are such that it appears to the programmer that the store instruction happened first and then the load instruction happened. The only thing you'll notice when you're typing the keys is that the program runs faster because the things that may cause long delays in modern processors are load instructions, that depending on what level of the memory hierarchy they have to access, may take up to hundreds of cycles to run. So running a load very early is advantageous because I can start that long wait on a long memory access earlier and be able to overlap other things with

that execution. But from the point of view of hitting the keys, all you'll see is higher performance.

THE COURT: So when you're talking about the programmer, the programmer buys this hardware that does all of this that the programmer doesn't see. The programmer puts in the program that he wants to accomplish, but then the hardware actually makes it work faster.

MR. DALLY: Right. So the programmer sees things happening in program order where the store happens first and then the two loads, and as far as the programmer knows, that's what's going on. And the hardware under the covers, as it were, is reordering things to get better performance by speculating that the loads aren't dependent on the storage and running them early. But the programmer or the person typing the keys doesn't see that. It's just a performance enhancement.

THE COURT: Right.

MR. DALLY: But it's a performance enhancement that has to observe the semantics of sequential execution as if everything happened in the right order.

Let me move ahead. So we've seen in our example how we can detect a conflict even with an instruction that hasn't completed and we don't have a conflict, in this case with one that has. So in the patent, what it

refers to is having load and store instructions be provided to this data speculation circuit so they can be entered in the table and their execution tracked, and that when a store instruction commits or retires, it checks its address against the address of all concurrent loads, all loads that are already completed but not retired, as well as all loads in stored execution but have not completed.

So this gets to the question of what it means to have in fact executed a load instruction, and in the claim, in fact executed is used from the point of view of the data speculation circuit. And so we're not using the term in its normal meaning, but we're using it in a very particular meaning. We're referring to the data speculation circuit, trying to ascertain whether a load has misspeculated with respect to a store.

So from the point of view of the data speculation circuit, that load has in fact executed if it has misspeculated, if it has started an access that is certain to return stale data because it's ahead of the store in the memory system, even if it has not yet completed its access.

THE COURT: But accessing is crucial.

MR. DALLY: Accessing is crucial to the behavior of the load. So if you're worried about -- for

example, if you're worried about the point of view of an execution unit, which is dependent upon that load to provide data, then the load is completed when it provides -- when it actually does the access and provides the data. But from the point of view of the data speculation circuit, it isn't worried about the data, all it's worried about is -- the whole purpose in the existence of this circuit is to look at the loads and look at the stores and say wait a minute, that load has misspeculated, you have to cancel it and rerun it. And so from the point of view of that circuit, the load is in fact executed as soon as you can determine that it was misspeculated.

THE COURT: And you can determine it by accessing the memory?

MR. DALLY: No. You determine it by comparing the addresses. As soon as you can see the store address and the load address are the same, that load has in fact executed before the store and should be canceled.

THE COURT: And don't you have to access the memory to see whether it's --

MR. DALLY: No.

THE COURT: -- the same?

MR. DALLY: All you have to do is compute the address. If we go back to our example of the load

buffer and the store buffer, we know that the load has misspeculated as soon as we know the store address. The load address we already know, because to issue the load we need to know its address to issue it to the memory system. The thing that wasn't known, the thing that comes in late, is the store address. But as soon as we know the load address and we see the store address, we can see that a misspeculation has occurred. We don't actually have to access the memory to detect the misspeculation.

Let me walk through the function of the invention. As you indicated earlier, the position of WARF is that the invention is three tiers, and this is described in the patent itself. And the first tier is that if a load has a clean record, if there's no history of this load having misspeculated in the past, it is assumed that this good behavior will continue and the load is allowed to go ahead and speculate, that is to execute ahead of stores without any further inquiry.

If on the other hand the load has a record, that record is recorded in something called a prediction table and the invention looks at the value of the prediction in the prediction table. In this case the value was 1, and that value was used as an indication as to whether the load is likely to misspeculate or not.

Depending upon the value, if the load will be allowed to misspeculate or not.

And then finally the third tier of the invention has to do with when to release the load if it has not been allowed to speculate; that is, if it's not allowed to speculate, the store that's in the prediction table in another structure called the synchronization table are used to say when the store that historically is loaded has misspeculated with occurs and is executed, then the load will be released and allowed to execute even if it's still speculative with regard to some other stores in the instruction window.

So here is a little road map of the argument. We've already seen that we detect misspeculation by comparing every store against every concurrent load, not just against certain loads that it's paired with. What we're going to do now is we're going to look at the second tier of the instruction and we're going to see that the decision to speculate or not with the load, to delay it or to execute it speculatively is made solely on the value of the prediction and doesn't take into account whether or not any particular store is in the instruction window.

So let's look at a particular case where historically the load A has misspeculated with respect

to a store X. The prediction table has been trained on this so that an entry in the prediction table associated with load A has a prediction that we'll use for the decision to delay the instruction and it is also recorded store X because that's the store instruction that this load has historically misspeculated with respect to.

So let's move on and see what happens under two scenarios. The first is when we encounter load A and store X is not in the instruction window, instead some other store, store Q is in the instruction window. So load A is still speculative because store Q hasn't completed yet and it may be dependent upon store Q. But the paired store, store A does not appear anywhere. And then we'll look at another scenario where store X is in the instruction window, and what we'll see is that the decision to delay the load does not depend on whether that store is in the instruction window. It will be delayed solely based on the value of the prediction associated with the load.

So here is Figure 3 of the '752 patent that describes the operation of the data speculation circuit, and the data speculation circuit here first checks to see if this is a load instruction, and it is, so it moves forward and says is this a speculative load. And

because there's an earlier store in the instruction window that we don't yet know the address of, it's speculative, depending on the address of this load/store. This load may be dependent on store Q or it may not be dependent on store Q.

So since it's speculative, it puts out a signal called handle ready to load. This is one of the signals from the data speculation circuit to the prediction circuit or the predictor. So here we're going to jump over to handle ready to load in the predictor circuit. In the predictor circuit we first check is the load in the prediction table. This is the first tier. We're seeing if this load has any record at all. If there's no record, we'll go ahead and just execute it speculatively. If it is in the prediction table, we'll go to Box 104 here where it says is synchronization required and the associated text indicates that that's determined by examining the value of the prediction that's associated with the load in the prediction table. It does not make any reference to the store entry in the prediction table. So that this will delay the load if this prediction indicates it should be delayed, whether the store that's made it speculative is store A, store X or store Q.

So the synchronization table is not needed for this

decision. Once it's determined that synchronization is required, the wait flag will be set to 1. This lower part of Figure 4 deals entirely with the synchronization table, and if synchronization is required and the paired store is not yet executed, the path will always follow down to Box 116 here, setting the wait flag to 1. And it's the wait flag set to 1 that will then cause the load to wait until it's either woken up by the paired store, becomes nonspeculative, in this case when store Q has retired, or if it's squashed because of the misspeculation or some other reason.

So the wakeup in this case, because the paired store, store X, does not occur in the instruction window, won't involve the synchronization table, it will just wait until it's no longer speculative, that is until there's no instruction ahead of it in the instruction window, that is a store with an unknown address, and once that occurs, the handle load signal will be asserted which allows the load to complete normally.

That was scenario one where we encounter the load, the paired store is not in the instruction window, some other store is, and the decision to delay the load is based solely on the prediction value. The identity of the paired store or the identity of the store that the

load is speculating on does not play in the decision whether or not to speculate on the load.

Let's consider the second scenario now where the store that's in the prediction table is the one or is one of the stores in the instruction window. Here the first part happens just as it did before. We determine it's a load. We see because there are stores ahead of it in the instruction window that it's a speculative load. We assert handle ready to load. We will look at whether the load is in the prediction table. We'll look at what the prediction value is in Box 104, and we will decide to delay the load. Wait will get set to 1 if the prediction is in the proper range, and that wait getting set to 1 will cause us to fall into -- go back one -- cause us to fall into this box where we wait for one of these events to happen. The only difference now is that if store X executes first before store Y, it will wake up the load and the load will execute even though it may still be speculative with respect to a different store, in this case store Y.

So here are the two instruction windows. The load here is making its decision on whether to speculate only using the value highlighted in blue here, the prediction value on the table. The value of the store that's paired with the load on the table is not used for the

decision as to whether to delay the load or to execute it speculatively. So the same delay will occur whether it's some other store or queue in the instruction window or whether it's the paired store, store X in the instruction window.

Thank you. Do you have any questions for me before I turn the podium over to Mr. Haslam?

THE COURT: I don't think so. Thank you.

MR. DALLY: Thank you very much.

MR. LEE: Mr. Steinberg, Your Honor.

THE COURT: All right. Mr. Steinberg.

MR. STEINBERG: Thank you, Your Honor. I'm Donald Steinberg. I'm going to end up covering some of the same ground I think that Professor Dally covered. We're largely covering the same concepts here, but I'm hoping to some degree a second perspective will help in the understanding.

There are four general areas I was planning to cover, and some of this I'm going to skip over fairly quickly, the relevant technology, dependency in load and store instructions. A little bit about what happened before the '752 patent, as Professor Dally indicated, there was a lot that happened before, and then try to get into the details and largely follow what actually happens and what's described in the '752 patent.

There are generally three types of instructions that -- I'm on Slide 5 -- are relevant here. There are computation types of instructions such as adding two numbers together, and then there are data-type instructions, which are the focus of this speculation that we've been talking about. There are store instructions, and those store a value to a memory location. And there are load instructions, which load a register from a memory location. So that's the data-type instructions. Then lastly, there are control instructions which jump the program from one part of the program to another part, and we're not going to be talking too much about those.

Generally the instructions appear in the memory of the computer and they're in a particular sequence or order and the patent calls this the memory order. So for example, in Slide 6, in the lower left we see the memory order. There's a load instruction, the LD. Then there's a multiplying instruction, and then a store instruction. And that's the memory order of the instructions.

But the instructions don't always execute in the memory order in which they're stored in the computer. One reason that may happen is because of the branch instructions I was talking about. They go down to

instructions 6, 7, 8, 9, then it branches up to instruction 4 or down to instruction 40. So if you have something like that occur, they're not actually going down in sequential order.

But perhaps more important for our purposes, what the patent is talking about is parallel processing and what parallel processing lets us do is execute more than one instruction at the same time. If you can do that, it speeds up the processing. Everything gets done much faster. But if you're going to do that, you have to add some extra controls into the process because things can get messed up and that's what the patent is going to relate to.

Now in determining what can be executed in what order, and I'm on Slide 8 now, it's important to consider both the concept of dependent instructions and independent instructions. So the example here is not from the patent, but I think it helps illustrate it a little bit easier. This was in the papers we filed. The second instruction on Slide 8 is dependent on the first instruction because it needs to know what the value of A is in order to multiply it by nine to get the value of B. If instruction 102 executed before 101, there would be a problem because it wouldn't have the current value of A.

But the first instruction, 101, is independent in this example because all it needs to get is 5 and 3. It's not -- it doesn't need some prior instruction to tell it. So that's the idea of independent and dependent instructions.

Now when we get into the patent in just a second, we're going to get into sort of a lower level of instructions because we're just talking about loads and stores, but the concept of dependency is the same. So we talked a little bit about load and store instructions.

Going on to Slide 10, here we saw Figure 2 before, I showed you the load and the multiplying and the store in the lower left. If a store instruction is going to store data in a memory location from which a later load instruction is going to retrieve the data, then the load is dependent on the store. The load needs to have the store execute and put its data into the memory location so the load can get the data. Until that happens, the load can't execute because it's going to have all data and it'll get the wrong results.

So at one level that sounds fairly simple. As we go on to Slide 11, the problem is that determining whether there's dependency is not always that easy because the memory location from which a load or a store

will work is not always clear upfront. And so really quickly in the example shown in the patent, and we're looking at Table 1 from the patent in Slide 11, the first instruction is store to the memory location that's identified by R1. The difficulty here is that until the program executes, we don't know what the value is in R1. And because we doesn't know what the value is in R1, we don't know what the memory location is where we're going to store this piece of data.

And then if we go down to the next instruction, it's going to load from the memory location that's identified by R2. We have the same problem. If we don't know what the value is in R2, we don't know where we're going to load the data from. So this creates an ambiguous dependency, because if R1 and R2 have the same value, then the load and stores are going to operate on the same memory location and then there's a dependency. But if R1 and R2 have different values, there's no dependency between these instructions because the store is going to store at one place and the load is going to read the data from some other place. So until we get to a point where we know what R1 and R2 are, there's an ambiguous dependency. We don't know whether the two instructions are dependent on each other. And so what we'll see as we go on, we're going to assume that they

might have the same value and so we're going to act according to that. And in particular, the difficulty here is that we have a choice when we have these ambiguous dependencies. We get to Slide 12.

If we execute the load out of order, that is we speculate in the terms of the patent, then the program will execute more quickly because we get to do the load in parallel with other instructions. But if it turns out the load was dependent on an earlier store and that store hasn't yet stored its value to the memory location, then we're going to have an error from executing it out of order. And if that happens, we'll have to reexecute the load instruction and that's inefficient.

So the other option instead of speculating is that we could play it safe and we can just execute all the instructions in order. Then we don't get any of the problems of misspeculation and wasting or having to reexecute instructions but at the same time we lose all the efficiencies of doing it out of order and letting the instructions execute at the same time. So we're trying to decide when should we go out of order, risk making a mistake, because that costs us something, or execute in order but then we risk slowing down the whole process.

And so the last part of this that comes in is that because there's the possibility that despite all our efforts there will be a misspeculation, that is a load will execute out of order but will turn out to have made a mistake, it'll turn out that it was dependent on earlier instruction, we have an extra step that we go through. So once we determine that there's a misspeculation, that is we find out that this occurred, there was an earlier store -- let me try to restate that. Once we've determined whether there's a misspeculation, we then know what we can do. If it turns out there was no misspeculation, so we proceed along and it turns out everything was fine, then we finish up the instruction and we can move the data, since it's a load instruction, into the appropriate register. If there was a misspeculation, even though we've executed the load, since we haven't totally completed everything, we can do what the patent calls squash the instruction, which really just means we disregard it, and then we're going to have to reexecute it once it's okay to execute it. So we've got this last check so that in case something goes wrong, we don't create an error that we can't recover from.

So I want to move into the patent now and just really fast, just to give a little bit of perspective,

as Professor Dally said, a lot of aspects of this existed before the patent. So out-of-order execution -- and this is all -- the patent readily acknowledges this. Before the patent out-of-order execution existed, this idea that you might take the load before the store. The idea of speculation existed. This idea that you might execute when you're not really sure whether there's a problem; in fact speculation with load and store instructions existed. And then the other thing that is talked about in the patent which also existed was predicting whether you should speculate. All of that existed.

So with that in mind, and that's all talked about in the patent, what then is the focus of the '752 patent? And if we look at the background of the patent, the background explains tracking all of the possible data dependencies, in the words of the patent, can easily become overwhelming. The program has lots of load instructions and lots of store instructions, so tracking all the combinations of loads and stores can be a very overwhelming task.

So the first paragraph of the summary, which is what we have up on Slide 17, explains how the inventors concluded that misspeculations tend to result from just a few load/store pairs. The summary explains that if a

load/store pair causes a misspeculation one time, it's highly likely the pair will cause further misspeculations.

So I'm just going to jump ahead a little bit. I think Professor Dally covered a lot of the parts of the apparatus, but just generally when processing instructions, just to keep in mind there are three things that the different processing units shown in Figure 1 do: They do computations, they store data to memory, and they load data from the memory into a register. And a load may be delayed, so it may not be executed out of order because there's a prediction that there's a high likelihood of a misspeculation. We're taking an educated guess that there's going to be a problem, and if we think that's going to happen, we're going to delay the execution of the load until we think the problem has gone away.

And we may squash load instructions if it turns out that we speculated, but that was a mistake. It was done out of order, so it was the speculation, and it was done erroneously. It turned out that the load was dependent on the store. If that happened, then we're going to need to reexecute it because there's a mistake here.

So the patent has this data speculation circuit, as Professor Dally indicated, and that's what detects

misspeculations. And just to try to keep moving along here a little bit, let me just move on to Slide 27. The other part that's going to be important is the predictor circuit. In the patent, the predictor circuit provides a dynamic indication, to use the words of the patent, of whether data speculation should occur. This prediction process that it's going through is going to change over time. And it's going to use this prediction, Slide 28, to determine whether to delay a load instruction or let it continue to go.

So to explain how the parts of the patent work together, I think it helps to start with Figure 2 of the patent. We've looked at that before, and jumping ahead to Slide 32, just to follow through one example of what happens according to the patent. So we have on the right side of Figure 2, it shows the sequence of instructions that are going to happen. There's a load, a multiply, a store, then we repeat that again. So if we just walk through that real quickly, and starting with Slide 32, the first thing that we're going to do in this sequence is we're going to load from one location in memory using instruction 8. And you'll see it says load A(1). So that's indicating it's going to load from memory location 1 for purposes of this example.

We then multiply what we just loaded from address 1

by 19, that's that second instruction 9, and then we get to the second highlighted one which is when we are going to store the results of that multiplication and we're going to store it in address, it says A(2). So think of that as the next location in memory where we're going to store the results.

And then we're going to get along, if we follow down to the next instruction of interest to us where we're doing the second iteration through the second set of instructions and we're going to do a load, but this load we see is from A(2). So that's the same place where we just stored it before, and we see in the patent it's got this arrow at 36, and just to highlight the fact that the store and the load are from the same location.

So if in any iteration of this process the load that's going to load to A(2) executes before the previous store, the one right there, I'm not too good with these arrows, so if the reddish store has not executed when we execute the load, there's going to be an error. So one of the things the patent is trying to do is to prevent these sorts of errors from occurring.

So one question may be why is it that the load may execute before the store? And we don't really know whether that's going to happen. The load and the store,

if we go back to Figure 1, there are these different processing units, and different instructions get executed by the different processing units so that they can execute in parallel. But depending on -- so it could be that the load and the store get allocated to different processing units as we go through the sequence. Well, even though the load comes later, if the processing unit that's handling the load is able to get through its instructions more quickly, then we could end up processing the load before the store. And what makes this particularly tricky is that it could be that the load gets executed first, but it could be that the store gets executed first, and in fact, if we do this several times, we may not always get the same sequence, because depending on what else is happening in the computer and how long some of the instructions take, it could be one time the load comes first and that's a problem, and the next time the store comes first and that's okay. So we're trying to deal with these possibilities.

So moving on to Slide 38, as we get into the heart of the patent, it's broken down, if we follow the figures into three groups of figures. Some of them deal with the data speculation circuit or the predictor circuit, and those are particularly relevant for our

purposes. And then there's some other figures which detail these prediction and synchronization tables, and I'll touch on those. But those details probably are not really that important. So that's just going to be a little bit of color to it.

So while mostly I want to talk about the detailed description, the summary provides I think a nice overview of Figure 4 of the patent, and since that's closely related to the invention, I want to start with that. Professor Dally talked about some of this a little bit earlier. You recall we talked about the first paragraph of the summary five minutes ago or something when I was talking about the inventors discovered that most misspeculations occur as a result of particular load/store pairs. So then after that we get to the second paragraph of the summary where they explain their approach. When you get to a speculative load, that is when there's a load and there's a prior store that hasn't executed yet, then their approach involves three questions. First, if there's no history of data misspeculation with that load that you're about to execute, then we're not going to delay it because that looks like a situation where this load, for whatever reason, is not going to create the problem. So let's just let it execute. We always have a backup. If

there is a problem, we can always squash it later, but for now let's just let it go. And that corresponds to Block 102 in Figure 4. Is the load in the prediction table.

Moving on to Slide 40, the second question is if there has been a misspeculation with a load, then the system looks at the history of misspeculations that corresponds to the prediction to determine whether the instruction should proceed to be executed or it might need to be delayed. This corresponds to Block 104, is synchronization required.

Now recall from the first paragraph of the summary the patent describes a prediction that's based on a load and a paired store with which there's been a history of misspeculation. I'll get into that a little bit more in just a minute.

So the third question, going on to 41, is if the answer to question two was that the instructions should be delayed because of a history of misspeculations, we get to this third question which is when should the load instruction be executed. And the next group of blocks in Figure 4 uses the synchronization table. We're not going to worry about the details of it, but that's what the summary states, to look at whether we've already encountered the paired store. And if we already

encountered the paired store, that is the one that we think is likely to cause the problems, we go down the left-hand branch of Figure 4 and in Blocks 120 and 122 we can execute the load now, and I'll get into those details in a little bit. But if we have not yet seen the paired store, that is the one that tends to cause the problems, we go down the right-hand branch and then we need to wait before executing the load.

So let me go through the structure of the disclosure in a little bit more detail. Jumping back to Figure 3 in Slide 42, the structure of the patent really centers on the data speculation circuit. And that's described in Figure 3. I'm not going to talk about all the steps, just try to focus on the ones that seem most relevant to understanding the patent. Professor Dally talked about a little bit of this, so I'm going to try to do this fairly quickly. But at Block 48, the circuit is going to determine whether it's a load instruction or a store instruction and take different paths, depending on what it finds.

In 43, we find that if it was a load instruction, then we look at whether it's data speculative. Now it's data speculative if there are prior store instructions on which the load may depend on its data. We don't yet know whether it's going to be -- whether they're going

to depend on it, so it might need to load data from a location to which a prior store will but has not yet placed data, and that creates the dilemma we were talking about before.

If it is data speculative, moving on to Slide 44, we move to the ready to load block, as shown in Block 70. That was what I was talking about just a moment ago that's referred to in the summary of the invention. So getting into the ready to load in a little bit more detail, Slide 45, the patent explains that the predictor circuit carries out this ready to load function when we need to determine whether speculative load should be -- should wait or it should be allowed to speculate.

So moving on to 46, what is it we do then? And we're focusing on Figure 4, and I've got Figure 5 as well. First, according to the patent, we determine whether the load is in a predictor table, and the predictor table we see in Figure 5 on the left and we've got that highlighted. It is described in the patent the load will appear once and only once in the predictor table.

Now in the example in the patent, moving on to 47, the load at instruction number 8 -- so we have load 8 is how they do that in shorthand -- is in the prediction table and it's paired with store and instruction 10.

What that pairing means is that there has been a prior misspeculation involving the loaded 8 and the store at 10.

So we know there's been, moving on to 48, we know there's been a past misspeculation involving the load in column 8 -- I'm sorry, the load in column 8 and the load in 10 in the second column, and then the third column of the prediction table is a value 1 in the patent. So the third column, that value 1 is a value that reflects the extent of prior misspeculations. That's the prediction value. So that's saying with loading and store 10, we have a prediction value of 1.

So moving on to Slide 50, the question we have to now determine is whether to allow this load instruction to execute speculatively. And we do that by looking at the prediction value and seeing how high is that number; in other words, is it sufficiently high that we think it's likely that there will be a misspeculation between this load and the paired store? Because if it's likely that there's going to be a misspeculation, let's not do it. But if it's not very likely, then the overall efficiencies of the system say we should just let it rest.

As the patent explains, the higher the prediction value, the higher the likelihood of a misspeculation if

the load in the first column of the prediction table is executed before the store in the second column.

So moving along in Figure 4 in Slide 51, if synchronization is required, that is we don't speculate, then we determine if the load arrived before its paired store; that is, the paired store is the one that made us conclude that we should do this synchronization, that is we should wait.

Slide 52. So if the load arrived first, we better wait, because if the load arrived first, that means we haven't yet executed the store. But if we do the load before the store and they've had a history of misspeculation, then that's likely to cause a misspeculation again.

On the other hand, 53, if the store already executed, that is the store arrived first, what we're going to do is we're going to set this wait flag equal to zero, and what that's going to do is it's going to allow the rest of the circuits to determine that we don't have to make the load wait because probably we won't have a problem. Remember this is really just a prediction because in the patent we're focused on this particular pair, the load 8 and the store at 10. So we won't wait in this case because even though the store has already occurred, there still might be a

misspeculation because of some other store. But we're focused on the load 8 and the store 10, and so if those two indicate there's no problem, we're going to let it speculate.

So because the store executed first, we're also going to conclude, and we see this reflected in Block 120, that the next time we see this load, since this time it worked out fine, we're going to say well, going forward, it's probably a little bit more likely it's going to be fine in the future. So we update the prediction, we make it a lower value because now we're a little bit more comfortable that we can speculate. Because we speculated okay this time, we decide it's a little bit safer to do that the next time.

So now we're back to Figure 3. We've covered Figure 4, the handle ready to load in Figure 4, and now we're going to use the results of the ready to load to determine whether we have to wait. We're now at Block 72. Basically the way this works is we're going to wait. We're not going to speculate if there's a high prediction value for this load and its paired store and we haven't yet executed the paired store in this iteration. So if there's a high prediction value, which means a misspeculation is likely with this particular pair, and we haven't seen the paired store yet, that

seems like a good situation to expect there's going to be a problem again. But if that's not the case, then we're not going to make it wait. If we decide not to wait, moving on to 55, we just issue the load request. That's the fairly simple situation. On the other hand, in 56 or Slide 56, if we need to wait, we're going to wait for one of three events before we can do more with the load operations.

Now we're in this wait state. The first one is that if later the paired store executes, since that store was what was the most likely to cause a misspeculation, we wake up the load and we let it proceed. That's the first one where it says wakeup.

The second possibility is that if the prior store instructions, if we find out enough about the prior store instructions to know that they're not addressed to the same memory location as the load, then it's safe to execute because now there can't be a misspeculation because they're not covering the same memory location. That's the second one, consumer no longer data speculative.

And the third one, the third possibility that gets us past this wait state is that the load might be squashed for various reasons. So I want to talk about the first two of these in a little bit more detail.

The first event is that the paired store executes, and in Figure 3, this means we go back through it another time. We have a store instruction as determined in Block 48, and so we proceed down through a couple more steps to the handle store at Block 64 and the handle store steps are going to let us get past the wait state and they're also what get used back up in handle ready to load in order to let us know that we've already seen the paired store. So unfortunately this patent is one where you need to see everything because they all sort of relate to each other.

So I want to talk a little bit about the handle store and this somewhat parallels what we saw in Figure 4 with the handle ready to load. At Block 201, if this stores in the prediction table and it has a history of misspeculation with this particular load, and if at Block 202 the prediction value is high, so we shouldn't let the paired load speculate because this load/store pair has a bad history of misspeculating, then what we want to determine is if we duck to the store before it's paired load. Because if we get to the store first, then everything is fine.

So at Slide 58, if the store arrived first, we set a value so that when we get to a load and it turns out we have now encountered the paired load, we will know we

already executed the store and we won't need to wait because we've executed them in the correct order in this case. So even though the prediction is high, because the store arrived first, it turns out our prediction was wrong in that case. We thought there was a high likelihood of a misspeculation, but the store arrived first, so we're going to let it -- we're going to let the load execute right away.

On the other hand, moving on to 59, if the load arrived first while we're in this handle store routine, that is the load arrived before the store, we're now in the middle of processing. That tells us we were correct to make it wait because the load arrived and then the store. That's the wrong order. That's what causes a misspeculation.

So if that happens, down in Box 214 we can update the prediction and say ah-ha, we were right. It reenforces that conclusion. And so we raise the prediction value saying it's more likely this trend is going to continue, but now since we've seen this store operation, which is the thing that causes the problems, since we've now seen it since we're handling that store now, we can now wake up the load and let it proceed. It's now probably safe for it to go. So that's this wakeup load and that corresponds to what we saw in

Figure 3, the first option when it was waiting to wake up the load.

So back to Figure 3 for a second, we've been covering how we get to the wakeup in Block 80. So briefly going through the second possibility, the second event that allows us to continue with the load, that could mean that the load is no longer speculative. This occurs when we've determined that the prior store instructions are for different memory locations than the load. So once we determine they're for different memory locations, it's not really speculative anymore because we know there won't be an intersection. It's not speculative, so we can now proceed to the handle load steps in Block 68, lower down in Figure 3, and those are described in Figure 10 of the patent and will actually issue the load request.

So I want to talk real quickly about the handle load steps and then there will just be one more figure to cover. We get to these steps, if the load waited, and now we've determined that there aren't any prior stores that may conflict with the load, that tells us it was a false alarm. We didn't really need to wait. Because we didn't really need to wait, we conclude that this particular load/store pair isn't quite as likely to cause a misspeculation and so we're going to update the

prediction to reflect that.

So lastly, we get to Figure 9, which looks a little ugly I think, but I'm only going to cover a few of the blocks in Figure 9. Figure 9 is the handle misspeculation, and this is what we do when there's actually been a misspeculation. So we speculate with a load, we had it execute, and then it turns out that some store that proceeded it was to the same memory address, and it may be the load/store pair that we're focused on, but it may be a different load/store pair that caused that to happen.

So in block or Slide 63 rather, if the misspeculation involved the load/store pair we've been following, that reenforces our belief that we shouldn't speculate. Because we speculated, we were in error, but this is the one that we think causes problems, we weren't really sure yet, so we're going to raise the prediction value so maybe we won't misspeculate the next time.

Slide 64. But what if the misspeculation was caused by a different load and store combination than the one we were following? Because that can happen, too. So if there's a pair that's in the prediction table, remember that there's only -- every load only appears once in the prediction table and the store only

appears once in the prediction table. So if there's a pair in the table that matches just the load, we're going to update the prediction to indicate that it's safer for that pair to speculate because it's a different load/store pair that caused the misspeculation.

So just to try to restate that a little bit, if the prediction value -- well, let me just finish up first. The same thing happens with a store. If the store is in the prediction table but not the load that caused this misspeculation, because misspeculation is always a load and a store in order to get a misspeculation, if the store is in the table, but not the load, we're going to decrease the prediction value for that, for the load/store that are actually in the table because a different load/store pair caused the misspeculation and the patent focuses on the one load/store pair that it thinks is most likely to cause a misspeculation. They say in the background it's not tracking all the load/store combinations.

So when we get a misspeculation, the prediction value might go up or it might go down, which at least to me seems somewhat counterintuitive. It goes up if the misspeculation was caused by the load/store pair that we've been tracking in the prediction table. It goes

down if it was caused by a different load/store pair than what we're tracking in the prediction table. You're looking at me as maybe that wasn't clear.

THE COURT: No, that's fine.

MR. STEINBERG: That's everything I wanted to cover. This last slide is really just here as a reference. I'm not going to try to cover it. It kind of summarizes at least in my mind what we've been talking about. It's sort of an annotated version of Figure 3, but I think we can continue on with the hearing. Thank you.

THE COURT: Thank you. Mr. Haslam.

MR. HASLAM: I have a set of slides. I doubt I'm going to use them all, but I will probably refer to some of them, so I've got a set. I'm also going to hand up, we didn't do it earlier, the tutorial slides that Professor Dally used.

THE COURT: Good.

MR. HASLAM: As is frequently the case, we agree on about 80 to 90% of the description of the patent and how it operates, but there are some fundamental differences and I think the fundamental difference laid out in the briefs is Intel's position that the most important load/store pairs in the claim will result in erroneous instructions being performed.

Our interpretation does not do that.

If I step back to the point of view that the Court asked earlier, which is let's look at this from the point of view of you at the keyboard, at least proverbially, when somebody writes a program for example to update account balances in a checking account, let's say the simple program is you take the existing account balance and add any deposits to it so it's going to be a load instruction and some add instructions, you care that it is always right. You don't care that it is sometimes right and sometimes wrong. And therefore what that says for this invention and the import of my argument is, contrary to what Intel argues in its brief, and I'm going to show you some of those sections in a moment, you must detect data dependence between a load and any store where they are executed out of program order because any store, as the patent says, could cause problems with that load.

Now the specification does talk about load/store pairs and the fact that load/store pairs do permit you, if you keep track of them, to determine which combinations are highly likely to cause misspeculations. But going back to the real world, you care not only about whether highly likely misspeculations occur, you care about whether any misspeculations occur, and the

patent Claim 1 is directed to that aspect of the invention.

Now I want to review just briefly some of the points that Professor Dally made because I think it's instructive to go back to that and then look at the arguments that the respective parties are making. One of the first points that Professor Dally made was that a load is determined, whether it's speculative or not, based on the presence of any stores in the instruction window, not just the paired store, but any instruction. And that we see, if we look at Figure 3, Box 66, will this be a data speculative load. That is the point in the flowchart where we have a load instruction which has come in, we get over to Box 66 and we want to say is this a data speculative load. Is this a load that may depend on other instructions, stored instructions in the instruction window. And if it is, we're going to do certain things with it, and if it is not, we're going to do other things. And the no box means if it's not potentially data dependent, we're going to go ahead and execute the load, go down to no, back to Box 68.

But let's look at column 10, line ten, and see what the patent says the decision at Box 66 is made on. The paragraph actually begins at line eight. If at decision Block 48 the instruction received by the data

speculation circuit 30 is a load instruction, then at decision Block 66 it is determined whether this is a data speculative load; that is, whether there are prior store instructions on which it might depend. Prior store instructions, any store instructions, not just the one that may be highly likely to cause the problem.

Then we see in the flowchart, since we're on column ten, I'll stay in the specification, what happens in Box 66 after you've determined that it's going to be data speculative? What do you do with it? You're going to make -- you're going to look at what the prediction for that particular load instruction is because it is the prediction associated with the load which then determines what you're going to do with the load. Are you going to delay it or are you going to let it go ahead and execute speculatively?

And if we look at column ten, line 19, it says "the predictor circuit will address the ready to load request from the data speculation circuit by making a prediction as to whether the load should take place through the use of a wait flag."

So what it's doing is, as it says there, it is looking only to see what the prediction with respect to the load is to make the decision as to whether it's data speculative. It does not make the decision as to

whether it's going to treat it as a data speculative load based on anything to do with the store that may or may not be present which may or may not be the one that is highly likely to cause the problem.

And as I will get to touch on later, but as Intel's counsel said during their tutorial, they noted that the prediction is updated regardless of which store caused the problem. In his point he mentioned he thought it was somewhat counterintuitive, but the important thing is that the prediction is modified, not just if the paired store happens or not. But if there is or is not a misspeculation based on any other store, the prediction may be modified, which again emphasizes, goes back to the point of view from the real world perspective here, this invention wants to make sure that it prevents any misspeculations.

If we were to walk through the rest of Figure 3, if we looked at Box 80, which is the box that indicates what happens if you're going to wait or delay the load or if you go out to the other side, the no side whether you're going to speculatively execute it and go down to Box 76, as everyone agrees, both of those boxes, 80 and 76, will not commit the load until it is checked to make sure that all the stores in the instruction window have not conflicted. Again, not just whether the paired

store has happened, but whether any stores have happened.

Now I'm going to come back to it because there is no doubt that the specification does talk about in many places load/store pairs and load/store pairs are important to an aspect of the invention. It's the import of our argument. I think the claim language and the burden of our argument supports it is that Claim 1 does not need nor talk about nor depend on the load/store pair. That is really about the third tier which is the synchronization, and I'm going to come back to that, but I don't want the Court to think that I don't recognize that the patent does talk about load/store pairs a lot because to get the maximum amount of this invention, you may want to use the synchronization circuit, which does depend on whether or not the load that's highly likely to have caused the problem has occurred. But that is not a part of Claim 1.

Now what is, in my view, the importance of the fact that loads are speculated on based solely on the prediction associated with the load, and the fact that you do not commit or retire a load until you have made sure that every store that could potentially conflict with it, not just the paired one, is executed? Can we

go to Slide 1?

What I've put up on the slide are some of the statements out of Intel's brief on which it bases its argument that the Court should import the limitation of load/store pairs into Claim 1. And it is the burden of Intel's argument that the '752, the purpose is to detect data dependence between a particular load and a particular store. And it says that time and time and time again, that that is the purpose of the invention. And it is not the purpose of the invention. The purpose of the invention is to make sure you detect any data dependence. Go to Slide 2.

THE COURT: If I understand what Intel's argument is, perhaps I think they're arguing that the only kind of misspeculation is that between a load and a store. So you could read that as you're looking for this particular store that's coming up and/or load that's coming up and checking whether there is a particular load somewhere -- I'm getting them mixed up. You've got the store, you're looking for a particular load, you've got a particular store. You've always got one of each thing that you're looking to see whether they will misspeculate or not, whether they will -- one will not allow the other one to come in.

MR. HASLAM: That is their argument, and if we

can go to Slide 2, this is out of page 35 in their opening brief, but if we go to the next slide, here is the argument and the example that Intel puts in its brief that states how they believe the invention operates and why they believe that you're only looking for the particular store and the particular load. And this is the example they have given. But I think the important point is the portion of brief that we've highlighted here. "The whole purpose of the invention is to disable speculative execution of the pair A-X, not A or X individually." And A in this case is the load, X is the store. What we've seen, and both parties agree, in their tutorial demonstrations, is that in fact you do care about load A individually. As we saw, the patent looks only at the prediction associated with the load when it makes a determination as to whether or not it speculatively executes the load. It doesn't look at the store to see whether the store was there, as we saw with respect to Box 66. In order to make the determination as to whether to speculatively execute the load, it doesn't just look for the paired store, it looks at all stores. And it has to do that because if it only looked for store X, which is the one that's highly likely to cause a problem, if that's all it looks for like Intel says and like Intel wants to put in the claim, what

happens, we all admit it can happen, Mr. Steinberg walked through and showed what happens if it's a different store that causes the problem, you'll miss that. If Intel is right, you will miss the fact that load A in this example conflicts not with X, but conflicts with Y. And that was the burden in part of the example that Professor Dally went through.

And here on this, on Slide 4 that we have up now, we've got the store Y, store X, load A, load B, and Intel says we are only concerned with looking for misspeculations between A and X. If we go to the next slide, here are the two scenarios that -- Slide 5 -- these are the two scenarios that Professor Dally went through. Scenario one, you have load A, but in the instruction window store X does not appear but there is a store Q. And store Q, if at the time you're deciding whether to speculate on load A, what -- both parties agree, what does the patent say it does? It looks only at the prediction associated with load A. Does this prediction indicate that this load is okay to go ahead and execute speculatively? Or does this behave sufficiently badly in the past that you're not going to let it speculatively execute, and that's associated with the load, and as Professor Dally indicated and as the patent says, you make that decision based solely on the

prediction associated with the load. And once you've made that decision whether you're going to speculate or not speculate, in both cases, whether you're going down Box 80, which is where we're going to wait and delay it, or whether you go down the other side to Box 76 where you're going to speculatively execute it, you're going to then not commit that until you've waited to see if all of the stores have executed or whether you've gotten to the point where you can determine that none of the stores are going to conflict with that load. Even if, as in scenario one here, store X, which is the one which was highly likely, which is the one they say you only care about, occurs or doesn't occur and that's the burden of scenario one and scenario two, in both of those cases where store X, the highly likely one is in the instruction window or whether it's not, you're going to look at the load, you're going to see whether or not the prediction for this load is its okay to go ahead and execute it or not based solely on the prediction for the load, and then you're going to wait and see if any of the stores, whether it's a highly likely one or not, cause a problem.

THE COURT: I wanted to ask you why in your proposed construction you have this tracks execution of such instructions? And I understand what you're talking

about, but I just wondered whether that was a limitation you're adding to the proposed instruction.

MR. HASLAM: It is -- I don't believe it's adding a limitation. What it is explaining is something that the specification indicates the data speculation circuit does to accomplish its objective which is it tracks the operations.

THE COURT: Is there any other way to do that, to track the miscalculations?

MR. HASLAM: In the generic sense of what tracking execution is, the answer is no. It has to somehow keep track of what's going on so at some point in time it can determine whether there's data dependence. In a sense, Your Honor, because load/store pairs is a significant argument, and that's what we're having here, if the Court agrees with WARF that load/store pairs is not something that the Court should import into the claim language, then one can argue that you don't need to construe data speculation circuit at all because the claim pretty clearly lays out what it is. It's a data speculation circuit. It receives misspeculations, and the claim goes on to say what is a misspeculation. It's basically when a load executes before a store erroneously or causes a problem. But it tracks the instructions so that it can determine whether

or not there's a data misspeculation or not.

THE COURT: All right.

MR. HASLAM: Now if we -- so I think as we've gone through the specification, I think the specification in the examples of both parties indicated, show that Claim 1 doesn't depend and that the patent itself has examples where you speculate or not on a load based solely on the prediction for the load and that you then wait to make sure that a store does not conflict with that load, any store. So I don't think you can necessarily come to the inescapable conclusion that load/store pairs has to be in Claim 1, and if we just look at the claim language of Claim 1, there is nothing in the claim language that mandates or requires, based on the language alone, which of course is the starting point and ultimately the ending point of any claim construction is construing the claim language, obviously in light of the specification, there is nothing in Claim 1 which indicates that the load/store pairs has to be in or is included within the claim language.

Claim 1 is a processor that has a data speculation circuit which the text data dependence between load instructions and store instructions that are in fact executed, we'll get to that, before the data producing instruction. So there all that says is it's a circuit

that's going to detect the data dependence between loads and stores. It doesn't say between a particular load and a particular store. It doesn't say the store and the load. It says a data consuming instruction, which is a load, and a data consuming instruction, which is consistent with what both parties showed you which is ultimately before it commits the load, it makes sure that not only does the highly likely one not cause the problem, but any other one doesn't cause the problem.

And then if you go down to the predictor which receives a misspeculation indication to produce a prediction, what does the claim itself say the prediction is associated with? It says it is associated with the particular data consuming instruction, which is the load instruction. And the prediction threshold detector just says you're going to make some determination based on a threshold, comparing it to the predictor, the prediction as to whether or not you're going to execute the load data speculatively.

So the claim language doesn't support Intel's argument. And there's a claim differentiation argument. I know the claim differentiation is not the be all and end all, it is a tool the Court can use and sometimes it applies and times it doesn't. But if we look at Claim 3, this is dependent on Claim 2. But Claim 3 says

"wherein the instruction synchronization circuit includes a prediction table listing certain data consuming instructions, loads, and certain data producing instructions, stores, each associated with a prediction." There in plain terms is the provision for load/store pairs which Intel wants you to insert in Claim 1 and we believe at least this is a case where claim differentiation is instructive and you should not therefore import from the dependent claim into the independent claim where the language doesn't require it and the operation of the circuit as described in the patent doesn't require it.

THE COURT: Was it your intent to talk about anything other than the speculation circuit?

MR. HASLAM: I will talk about in fact executed. I can do that or if you'd like to go point/counterpoint, Mr. Lee can get up and address the load/store pair issue and I can get up and address the --

THE COURT: Okay, we'll take ten minutes at this time, and then we'll return. Mr. Steinberg, will you be taking the lead?

MR. LEE: Your Honor, I'm going to do it.

THE COURT: Okay.

(Recess 10:31-10:45 a.m.)

THE CLERK: This Court is again in session. Please be seated and come to order.

THE COURT: Mr. Haslam, were you finished?

MR. HASLAM: I just have two more points and then I'm done.

THE COURT: Okay.

MR. HASLAM: If we -- I just want to look for a moment back at Figure 3. We've talked a lot about, going down the right-hand side, which is what do we do with load instructions when they come along. I also think it's instructive on this point of whether or not, the significance of load/store pairs, and whether that's all we care about or not. If we look at the left-hand side, what happens when a store request does come along. So that after Box 48 where we ask is this a load or a store, let's look at what happens when the store comes along, any given store.

You issue a store request, and then at Box 52 you detect a misspeculation. It's the burden of Intel's argument that the only misspeculation you're checking for is to see if the pair store happened with respect to a given load. But if we look at the specification, column nine, and I've got a slide on this, Slide 6, if we come down the left side of Figure 3 and we get to the misspeculation Box 52, the specification says "at

decision Block 52, the data speculation circuit 30 checks other concurrent load instructions." So any load instructions which are sitting there in the instruction window, it checks. The store checks other concurrent load instructions to see if they have been prematurely executed and thus whether there has been a misspeculation.

So this sort of approaches it from the other side. We saw that the load has to wait -- that we've made the prediction and decided whether or not to delay or speculatively execute has to wait for all the stores, and we also see that the store gets checked against all the other loads to see whether there's a misspeculation. And that store can be any store. It's not just the store that's paired or that Intel says would be paired with a particular.

One final point I'd like to make goes to there was a lot of discussion -- some discussion by Professor Dally and a lot of discussion by Mr. Steinberg on the synchronization circuit. And if we can go to Slide 11 for a moment, I've put up here the three tiers that are referred to in the specification at column three, beginning at line 63, and I don't think there's any dispute on this, the first tier, there's no dispute of speculation. Go ahead and issue the load. If it's been

predicted to conflict, you delay it. And the third tier if delayed, then you may use a synchronization table. One of the burdens of our argument is that the third tier, the synchronization table, is another feature of the overall invention. But it is not a feature that is claimed in Claim 1. And Intel agrees with us on that.

So if we can look at Slide 6, for example -- I'm sorry, Slide 12. Here out of Intel's brief, the '752 describes a synchronization table that is used in conjunction with the prediction table to disable speculation. And then it says, if we look at Figure 4, for example, elements 106 through 122 and accompanying text really are not relevant to the parties' dispute here. This is not relevant to Claim 1. It's relevant to the synchronization circuit, which again is added in Claim 5 and some of the dependent claims, and that is a further optimization, the purpose of which is if you decided to stall a load, because it's got a prediction that suggests that it's prone to misspeculation, there is an added advantage if you want to make a further bet or a further guess that the only problem you may have with it is the one caused by the highly likely store instruction. And then you can make a further bet which says as soon as that instruction comes, I'll let the load go ahead and speculatively execute because I'm

going to bank only the highly likely one is the one that's going to cause me a problem. But even in that case, the patent says you still have to wait before you commit even that operation until all the stores have happened. But the synchronization table and the synchronization circuit is not the subject, as we all agree, of Claim 1. And while that -- what that tells us is while the figures and the specification talk in sort of global terms about an embodiment and the figures, the parties agree that portions of the specifications and portions of the figures are not relevant to the issues we're talking about in Claim 1. Yet in the tutorial, and I suspect in a lot of the arguments, Intel will be relying on those portions of the specification and drawings which relate to the synchronization circuit and the synchronization table that we don't care about in Claim 1 in order to support their argument. Thank you.

THE COURT: Thank you. Mr. Lee.

MR. LEE: Thank you, Your Honor. We have a set of slides that go just to claim construction, your Honor, and I'm going to start at Slide 17. I think Mr. Muller has it now.

MR. MULLER: These are different, two copies of the same thing.

MR. LEE: Can I proceed, Your Honor?

THE COURT: Yes.

MR. LEE: Your Honor, moving directly to this question of the pairs and what the patent says, let me just start with one legal proposition which is from the Phillips case at page 1319. And I start there, Your Honor, because one of the key concepts that Professor Dally communicated to you today was the buffer, what buffers did. You will search the patent forever and not find the word buffers in the patent. You also will find the word tier in the patent once, and I'll come to it in a second, not making the allocation that Professor Dally and Mr. Haslam rely upon. In fact, not making the synchronization equals tier three argument that they now make, and in fact describing what the circuit does in a manner which is very consistent with our claim interpretation, what Mr. Steinberg has presented to you.

And the key part of Phillips is the public notice function of the claim. And I think when you're presented with a claim construction, it's dependent upon tiers when it appears once in the claim, or dependent upon an understanding about buffers whenever it appears in the patent. The question of what the public notice -- whether the public notice function is critical, and if I go to Slide 17 and the issue Mr. Haslam just addressed, I think this maybe crystalizes the dispute.

If Your Honor looks at Claim 1 in the portion we've been talking about, WARF's principle argument is that we're trying to import a limitation into the claim. That's not true. There is a basic principle of law, your Honor, that if you use the word "a", the article "a", and you follow by using the word "the", that the "the" refers to whatever was described with the "a", and I'll identify a couple of cases where the federal circuit has explicitly so held for Your Honor. But first let me try to show you why it's important.

If you look at the portion of the claim that begins "a misspeculation where a data consuming instruction", so if I can do this right -- did I get it right? I'm getting some instructions from Mr. Braun here. All right. So Your Honor, having mastered this part I hope, you will see the first thing referred to is "a data consuming instruction." What kind of data consuming instruction is it? It is a data consuming instruction dependent on its data for --

THE COURT: Vice versa.

MR. LEE: Yeah, a data producing instruction. So you see that there is a data consuming instruction and then there follows a data producing instruction and the claim itself explicitly describes the relationship, which is they are dependent. But the claim then goes on

and it says that, if I go down a little further, is in fact executed before the data producing instruction. So what's described in the claim itself is the data speculation circuit. The purpose and function of the data speculation circuit is described, it's explicitly described as detecting a misspeculation where a load instruction is dependent upon a store instruction and the load instruction is in fact executed before the data producing instruction.

THE COURT: But as I understand it, there's always -- you're always looking for a store instruction. You have a load instruction and it will be executed only if there is no conflicting store instruction.

MR. LEE: It --

THE COURT: But then there's another aspect of the patent it talks about. We've seen that these particular combinations of load and store will often, always, highly likely produce conflicts.

MR. LEE: And Your Honor, that's exactly true. That's what Mr. Steinberg described in Figure 7, and there are three different situations the patent describes, but the resolution of them is all dependent upon the pair that's described in Claim 1. The first one was where the load instruction arrived before the store that has caused the misspeculation before. So if

I have, and for me this helps me understand it. If I have load instruction one, it's been executed before with store instruction one and it resulted in a misspeculation, then the patent says that okay, we're going to have this table that says load instruction one, store instruction one causes a misspeculation. And you count them. If there are more misspeculations, it goes up. If there are fewer, it goes down.

And let's just say for the purposes of discussion the threshold is 10. You've said if I get above 10, I have a problem. What the patent says you do is when the load instruction comes in, you look to see if it's in the table that has the pair and whether you're at ten or not. If it is, and you're over my threshold of 10, then you wait and you wait for three instances, and that's why Mr. Steinberg took you through the wakeup, no problem, squash. And what are you waiting for? First you're waiting for the store to arrive.

So in my example of load one, store one, with that being the problem, you delay load one from performing its function and you wait for store one to arrive. When the computer says store one is here, you can go ahead and your bank account is fine. I agree with Mr. Haslam this is all about getting it right and getting it right without misspeculating, having to do it over and over

and over again.

The second scenario is okay, I know that load one and store one caused my problem. I'm looking to see if store one arrives. It never arrives. Instead stores 6, 7, 8, 9 and 10 arrive and that is what load one is using. Then I know based upon my pair in my table that I shouldn't have a problem because the pair that caused the misspeculation isn't one of the pairs and that's why. What they've done is they've gone to the 60,000 foot level, and what I think Mr. Steinberg tried to do is take Your Honor into what the patent is really talking about. Because if you look at what the patent is doing, which is saying we've got this table and we're going to have three situations: Situation one, we'll wait for store one to arrive; situation two, we'll wait for all the stores to arrive and then we'll look at them and say store one is not one of them. We don't have a problem. And then the third is something else has happened in the system. We don't want to perform this load instruction. We're going to squash it. And those are explicitly what's described.

If you take that portion of the presentation, I don't think there's any real disagreement on it. It's just that I think we focused on it maybe a little bit more. The claim and the specification make sense, Your

Honor. Then the terms I'm focusing on in Claim 1, a data consuming instruction, a data produces instruction, a load to store, and then referring to the data producing instruction, and if I eliminate my highlighter here, it's the data producing instruction. It's been executed before the data producing instruction. That word "the" is critical. Its importance probably only appeared in the last round of briefing.

But in the Warner Lambert case which is at 316 F3rd 1348, 1356, the federal circuit held something it's held consistently which is if you use the word "a" to refer to something, then you refer to the same thing with the word "the". The "the" is referring back to the "a", and as the federal circuit said, it's a word of limitation.

So first the idea that we're just trying to take the word pairs and import it in is not correct. What we're doing is saying you can describe pairs in a number of different ways. I can say I have a pair of shoes. I also can say I have a left shoe and I have a right shoe, and all this claim does is say I've got a load instruction, I have a store instruction, and the misspeculation is determined by execution of the load before the specific store.

So I can go to Slide 18 and the issue we're talking about now is only the first issue which is should the

proper construction of Claim 1 refer to the load/store pairs? And we would suggest yes.

Going to Slide 19, the question is why. And Your Honor, there are really three reasons. The first is the focus needs to be, we would urge, on the intrinsic evidence, the claim language, and the specification. And if we focus on the claim language, it may be that both of us in our effort to try to focus the issues have done the Court a disservice. In some sense what we're fighting about is not so much what a data speculation circuit is, we're fighting about those words "a data producing instruction", "a data consuming instruction", and the word "the". And those are what's critical.

Here quite apart from the claim language which makes perfect sense if Your Honor considers what the specification actually describes, we have more. We have a single embodiment of the patent. There's only one. And we have a specification describing key portions as the invention. And one of the things the federal circuit has done post-Phillips is to give the specification a little bit more life and importance. And if you use the word "the present invention" to describe generically what you're doing, it has given it more import.

And if I go to Slide 20, these are the portions

which Mr. Haslam candidly conceded are in the patent. And what's described is the central insight, and at least as I understand the central insight, Your Honor, is if I have 20 loads over here and 20 stores over here, detecting all the different permutations and combinations that could result in misspeculations is a lot of work and in fact, the patent uses the word it's overwhelming. So what they say is here is our solution. Rather than having a method for detecting misspeculations that requires -- rather than either having no method for detecting misspeculations which would require just to go in order or having a method that allows you to speculate as much as you want and then squashing every mistake and starting over again, here is what we're going to do. We figured out, and this is why the summary of the invention in the first paragraph is so critical, that there's only a discrete number of pairs that causes the problem. And that's the first sentence, Your Honor. "The present inventors have recognized that most data dependence misspeculations can be attributed to a few static store/load instruction pairs." And it goes on several times.

But if I were to pause there, Your Honor, and go back to Mr. Steinberg's example with the three different scenarios, that sentence makes perfect sense in light of

what's described in the patent. We believe the inventors say that of the many different permutations and combinations that invoke from my ten stores, my ten loads, there are a discrete number, three, that are going to cause all the problems.

So what do I do? I figure out which of those three are by counting up my misspeculations. I put them in the table, and then when I go to the load instruction that's causing those, I figure out if the stores come first, if the stores are relevant, or it's squashed for another reason. It all makes sense and it doesn't divide the invention into the tiers that WARF has urged you to divide it into, and I think I can show that in just a second.

THE COURT: What I don't know is that is there a problem -- is there only one kind of problem that a load will face? Does it face a problem only when there's a store with the same number?

MR. LEE: Your Honor, let me try to answer it.

THE COURT: You can check all these different things, but is the only thing you're looking for to see whether it has the same number?

MR. LEE: Let me try to answer and then also say Mr. Steinberg is both more qualified and smarter than if I am, so if he nods to me and says I've got it

right, it would be right. If he says that I'm wrong, I'll let him answer. Is that okay?

I think in the manner in which it's described in the patent, the problem that you're having is that a load wants to load information and it's loading it from a store and it has not yet arrived. That's the example, the simple example that Mr. Steinberg described. That's like a static moment in time. If you think about the program running --

THE COURT: I'm here with a moving van, ready to pick up the furniture, and the owner isn't there to open it up, let me move this furniture. So that's what I'm doing. I'm saying to my computer find X and it's going to search around for X, which is in store somewhere.

MR. LEE: Right. Or the example I've thought of is I go online and I like to know what the balance of my checking account is and it goes to load the information on my account and compute what the net is today, but it hasn't -- you know, it goes to access information about the last deposit from my firm, it hasn't made it there. If it loads the information, which is zero, it's going to come out with a wrong number. If computing my check balance, which again I agree with Mr. Haslam is important that it be right, if

computing my check balance needs to be right, having it do the computation of my net balance before the information has arrived about my weekly paycheck is going to result in something that's wrong.

Now I think that if you removed yourself from that static moment in time and think about these programs running over and over and over again, millions of times a second, it may be that that same load instruction at some later point in time will need to load something from a different place. So that in my example of my checking account, it may be that the next time through it needs to load the information from the last check I wrote in order to do the computation.

So the way you describe it is exactly right which is it's a little bit as if you arrive with a moving van but the house is locked and you need someone to unlock the house so you can start moving. But the next day it may be that you'll drive your moving van across town to move someone else. But the same thing is true, you're going to want the house to be unlocked before you can move. And what the patent is saying, this is the summary of the invention, it's the only embodiment. What the patent is saying, you know, from all the different problems that can arrive from my moving van arriving at 100 houses, I know that there are only five

of these people who in this neighborhood are crazy enough to lock their houses all the time and I have identified them, right? And if I then get an instruction that says let's move this house and I know this caused a problem before, what I do is I look at my table and it says this instruction to go to this house has been a problem at 142 Forest Street.

So what do you do? You do one of three things. You wait until someone comes to 142 Forest Street and unlocks the door, then you're off to the races. That is the store arriving. What else can you do? You can wait until your whole manifest is in for the day and you know the five houses you have to go to and none of them are 142 Forest Street, and so you know okay, I'm okay to go because I'm not likely to end up at a house that's locked. The other is someone says hurricane in town, we're not moving anybody. We're squashing the instruction for you to go. That's the three scenarios the patent describes. That's what Mr. Steinberg described in Figure 7. That's why the word pairs is so frequently in the patent. And frankly, that's why no one has ever climbed the tiers in the claim in the patent itself and no one has ever equated synchronization with only what Mr. Haslam and Professor Dally described as tier three.

THE COURT: When you say no one has done it, who would you be talking about?

MR. LEE: I would actually be talking about the patent. I misspoke. That's why the patent never describes it in that manner. Because if you take what the claim says and then trace it to what the specification says is the heart of the invention and then trace it to the figure as Mr. Steinberg describes, it actually makes sense to hold it all together. You understand what they're trying to do, you understand how they claimed it, you understand how the particular examples were.

And if I go to Slide 21, without going through them all in detail, we have set forth, and that's one of the reasons that Mr. Steinberg took the time to go through the details of the figures in the specification rather than a figure that's not in the specification, and each of them describe precisely what I think I tried to describe to Your Honor. So this is a question of reading that, as I said, we may have done the Court a disservice by focusing just on data speculation circuit and we're all focusing on an entire portion and the claim language refers to up there, it makes the misspeculation dependent upon "the pair". It does describe it as my left shoe, right shoe rather than a

pair. The specification then says here is what's critical about it, and every single example is built off pairs.

Now if I go to Slide 25 which deals with some of the arguments that WARF has made and Mr. Haslam has provided this morning, the argument on pairs and this idea of the three tiers and third tier not being in Claim 1 we would respectfully suggest isn't correct, and I think I can explain to the Court why. Also the idea that there is a situation that where the load is delayed, even if it is impaired with a particular store somehow validates WARF's construction also is not correct. In fact, the example I just gave Your Honor with the manifest with the five houses is what demonstrates it's not.

So if I go to our response, first the argument that WARF makes, Your Honor, basically I would suggest doesn't give credit to the language of the claim as described and how that language should be read in light of the disclosed embodiment and in light of the constant reference to pairs and how they're used.

This theory about --

THE COURT: What still stops me is that pairing is inherent in the whole process.

MR. LEE: Right.

THE COURT: Because you're always trying to see whether there's something there to unload the load of, so you've always got to make a match. So that wasn't any big deal to say that we're talking about pairs.

MR. LEE: Yeah.

THE COURT: But this very special thing that WARF is talking about which is the pairs that are so well-known to cause problems as to be separately identified.

MR. LEE: That's exactly right. Your Honor, I think I can explain maybe a little bit more clearly why you're exactly correct and why that would make our claim interpretation correct. If you imagine my ten loads and my ten stores, one of the things you could do by the brute force way is for every -- for load one, there be ten different possible loads with ten stores, and you go all the way through, so there would be 100 different possibilities. You could have a table that had that 100 different possibilities and you could count up. You could keep a record of every time one of those pairs misspeculated, and then when your load came, you could try to figure out from that large amount of information whether you had a problem or not.

What the inventors are claiming is from that group of 100, we figured out there's only four or five. There

are four or five pairs. That's why the table has them as pairs.

Your Honor, the best indication I can give you that's what they're talking about and that this tier structure, if I go to Slide 27, is not what they're talking about is this: As Your Honor considers this issue, we would ask Your Honor to consider page 40 of the WARF response brief, the one most recently filed, and let me read a quotation. And this is to support the argument of the tiers. What they say in the first full paragraph is this: "Similarly to practice only Claim 1 and the first and second tiers of the invention, a skilled person would also visualize an abbreviated prediction table." This is from Professor Dally's supplemental declaration. "Because the identity of a store instruction is not required to practice Claim 1 and the second tier, such a table would consist only of two columns, one containing a load instruction and the other containing the associated prediction."

Now I think the question we would ask the Court to ask itself is this: If Claim 1, really just Claim 1 and 2 as articulated by WARF and it's described in the patent and it's satisfying the public notice function, why is it we have to visualize the chart that would make that happen? And the answer is that's not what's

described in the figures. There's a reason that you have to visualize this chart rather than turning to the patent and seeing it and it's because the pairs are critical to delaying the load under any circumstances.

There's one thing which Mr. Steinberg said which I don't think is something Professor Dally disagrees with which is this: For every load instruction, it's only in there once at one time. So load one is in there when store 10 created the problem. It's not in there multiple times, it's in there once. And that pair remains key. Why? Because the inventors believe that they have been able to focus upon the ones that are causing the problem.

So the tier structure would come down to one paragraph. If I go to Slide 28, it comes out just one paragraph in the patent at column --

THE COURT: 3.

MR. LEE: Yeah. You're ahead of me and I don't have to get to that. So if I go to Slide 29, what I'd like to focus the Court on is what the patent says rather than a characterization of what the patent says. Actually we have a little bit of a recording from WARF's brief which says "in the second tier of the invention, a load instruction is delayed when a prediction associated with a load indicates a high likelihood of a conflict."

Now the words "in the second tier of the invention" are not in the patent. What Mr. Haslam has done -- what Mr. Haslam has done is taken the three "if" clauses and broken them down. But if we take that as the articulation, a load instruction is delayed when a prediction associated with the load indicates a high likelihood of conflict. Your Honor, that prediction only occurs based upon a misspeculation with a pair. How do we know that?

If I go to Slide 50, there's a quotation from column four, line 8 to 20 of the patent itself. Specifically the present invention provides a prediction associated with the particular data producing/consuming instruction pair. So what WARF has argued to Your Honor is this: There's three tiers and the issue of the pairs is never joined until tier three. How do they get there? They say well, the issue of the pairs is irrelevant until you get to the synchronization portion.

Now what we did say is the pairs are relevant to synchronization. That is a step further down the road. But what we never said in our brief is that that doesn't mean that the pairs are not relevant earlier on in the process. And what WARF has said at column four, lines 8 to 20, is "the present invention provides a prediction associated with a particular data producing and

consuming pair." So if you then take that and go back to column three and you look at what WARF is relying upon, "a load instruction is delayed when a prediction associated with a load indicates a high likelihood of conflict", that makes sense. But it only makes sense if you use the word prediction in the manner in which it's consistently used in the patent. It's always used, Your Honor, to describe a prediction based upon a pair. And in fact, Your Honor, if it wasn't based upon that, the wakeup, wait for everybody's address, the squash won't make sense. That's a division that's based upon the fact that there is a pair.

Now Mr. Haslam said at the outset that the crux, if I turn to Slide 30, that the crux of the dispute is the following: They have attributed to Intel the argument that -- what they have said is a load can be delayed even if it is not with the same store with which it is paired in the prediction table. That's true, but it's irrelevant to the claims construction issue that's before Your Honor. It's precisely what Mr. Steinberg described by three scenarios.

If I go back to my moving example, it's wait for someone to unlock the house, wait until I get a list of every house I have to go to, or wait until somebody tells me I don't have to move anybody today. But it's

all dependent upon pairs. It's not suggesting that my mover has to only go to 142 Forest Street. But it's suggesting that if they are going to 142 Forest Street, it's going to be a problem. You're going to get there, it's going to be locked, you're going to have down time, let's wait. And that's what the patent describes and the pairs.

And that's why Table 5 is so illuminating. The pairs is what tells you, if we can go to Table 5 -- Figure 5, I'm sorry, that's why Figure 5 is what it is. That's why Figure 5 has three parts and that's why there is no table with just LD8 and 1, and that's why WARF says you would need to visualize.

If I go back now to Slide 31, the key to unraveling the argument that WARF has made is this: It's in the patent itself and not in an amplification of the patent. A load is prevented from speculating only if it appears in the prediction table with a prediction value above the threshold. So it needs to be in Figure 5, it needs to be above my hypothetical 10. But a load has only been placed in the table and assigned a prediction value because it has repeatedly misspeculated when paired with a store. It's not, Your Honor, that I have load 1 and store 10 and then the first time it misspeculates I say 1 and then you have load 1 store 8 and it misspeculates.

Instead of taking that up to 2, it takes it down to 0 because the focus is on the pair, right?

So if you think about the invention, which is let's focus on the pairs most likely to cause a problem, every time it causes a problem, you drive the number up. Every time it doesn't, you drive it down. What Mr. Steinberg said is a little counterintuitive is this: If there's another pair that causes a problem and not this one, you drive the number down for that pair. Now at some point in time, at least as I understand the patent, it may be that your second situation becomes the real problem, not the first, and then the tables change, and that's what they're talking about and that's what the balance is.

And that's why, if I load up Slide 32, Slide 32, Your Honor, is the situation where I've got my load, it's causing a problem, it's above my threshold, and the system says stop, don't speculate. But then what happens is the reason it's delayed, and the patent tells us this, is because there have been problems with that pair before. But then all the store instructions come in and what I find out is I didn't need to worry about my pair so I can say go ahead. That's my second example in the moving situation.

THE COURT: So wait, if you've got A showing up

and it's had problems with X --

MR. LEE: Yes.

THE COURT: Okay. So then it's held until you find out whether X is in the picture or not.

MR. LEE: That's exactly right. You can do it one of three ways. You have A and it's caused a problem with X before and it caused it enough times that your counter is above the threshold. Then you can do these, one of these three things. You can wait for X to arrive.

THE COURT: Or not.

MR. LEE: Right. Or you can wait until you know that X isn't going to arrive. That's the scenario two. Or you can wait until things get shut down for some other reason. But it's all based upon A and X in the first instance and why the pairs are.

The second example, Your Honor, or not is sent by a false alarm, which is we thought it might be this pair. It isn't. We thought we were going to go to 142 Forest Street. We're not. So if we step back from it all, and go to Slide 33, the system claim -- this is not about importing limitations in the claim. The system claim described revolves around tracking these critical pairs. The data speculation circuit revolves around predicting these critical pairs.

If I go to Claim 38, page 34, the second central point is that an individual load instruction won't be prevented from -- will be prevented from speculating if the prediction value of its pair is too high. Your Honor's A and X, it's over 10. But there can be false alarms.

Let me just address one other point which was Mr. Haslam's claim differentiation argument. This is an argument that's founded on a premise which is, we suggest, not correct. The premise is that A and X are only relevant to synchronization, and because Claim 3, for instance, or Claim 5 and 9 refer to synchronization, that's claim differentiation. The premise is incorrect, Your Honor. The pairs and the prediction are relevant well before prediction in every single figure in the patent. They're the only example. And because they are, they're not differentiated.

And I think the really interesting thing is you won't find the word pairs. In the claims Mr. Haslam says implicates pairs, you won't find that word there either. So they're making a claim differentiation argument based upon a false premise and actually what the claims demonstrates are these two things. They demonstrate that the inventors use words other than the single word pair to describe A and X and the way the

claim is written is A, X, and misspeculation because X didn't arrive.

That all makes sense in face of the claim. The claim differentiation, there are no claims that describe pairs in any different way and the claim differentiation, Your Honor, is premised upon the idea that synchronization is the only portion of the patent that implicates the pairs is simply wrong, and columns three and columns four, which in column three the second "if" is described, as I understand WARF's argument, one of ordinary skill in the argument would have read that second "if" to be in tier two, but that second tier requires a prediction which column four tells you explicitly is based upon A and X. Thank you, Your Honor.

THE COURT: Thank you.

MR. HASLAM: Just a few brief rebuttal.

THE COURT: Okay. I do want -- we have 33 minutes left and I do want to ask a few questions and I do want to get into in fact executed because I am struggling with that.

MR. HASLAM: I understand. Just a couple points. One of the arguments Mr. Lee made is well, the claim language in fact has pairs because it talks about a data consuming instruction and a data producing

instruction which are dependent on each other, and he uses the second "the" means there's only two instructions. Well, that doesn't prove anything. I mean both parties agree that loads can conflict with any given store, so it can conflict with a highly likely store. But if a highly likely store isn't even there, it can still conflict with some other store.

So there may be a pair in a sense that is determined once you find a conflict or a dependency between them and it doesn't get them to the point where they say it is any particular store which is what that pair is about. What they're saying is well, the patent talks about a load and a store which conflict and therefore they must be talking about a particular load and a particular store, but we saw that the patent pays attention to and the prediction is updated regardless of which store causes the conflict.

Insofar as the tiers go and insofar as -- I don't think it's a misreading to use, when the patent itself says the three-tier approach and you look at the rest of the paragraph and they use the if, if, and if, that those are the three tiers it's talking about. And the second tier, I'm not sure if the import of Mr. Lee's argument was we were somehow misquoting, but at column three, line 67, the second if, which we believe is the

second tier, if there has been a misspeculation with a given load instruction, a predictor based on the past history of misspeculations for that load instruction, doesn't say that load instruction and its paired store instruction, it says "history of misspeculations for that load instruction is employed to determine whether the instruction should be executed or delayed."

Now Mr. Lee went on to quote a portion of column four, line -- I think he started at line 15 which he says supports him, and it does talk about a data producing instruction and a data consuming pair. But I don't think that one isolated statement of what the invention is necessarily means that all of the embodiments and all of the claims cover all of the aspects.

If we go down in column four to line 31, the federal circuit has said you have to be careful about how you use summary of the invention to either impress or not impress terms. But if we look at column 31, it says "thus it is one object of the invention to provide a predictor circuit that may identify data dependencies on an ongoing dynamic basis. Recognizing that there are relatively few instructions which will cause data misspeculations, these instructions are identified by reference to historical misspeculations associated with

the instructions as stored in a prediction." That's a reference only there to the instructions that cause the problems and the instructions that cause the problems are loads that are speculative and that happen to cause problems because a store gets them.

So there, if we want to argue what the summary of the invention says, there's something in there for everybody, but I don't think you can come to the inescapable conclusion that in Claim 1 based on the language of that claim that the only thing is is that it has to be a load and a particular store.

And finally, I made this point, but Mr. Lee time and time again in his examples, he referenced the wakeup call and how the wakeup call was very important because the wakeup call occurred when the paired store that he was talking about came along and you wake the load up because now you know it's highly likely the store has gone by. Time and time again, if you go back and look at the transcript, he referred to wakeup call. The wakeup signal is generated in column -- sorry, in Figure 4 -- I'm wrong, it is in Figure 7.

Figure 7 deals with the synchronization circuit, and as we showed in my opening presentation, they don't disagree. The synchronization table is not relevant to Claim 1. They called it in their brief an

implementation detail which is not relevant to Claim 1. Well if it's not relevant, if the synchronization circuit isn't relevant -- synchronization table isn't relevant to Claim 1, which is in Box 204, then the wakeup call which comes after that which the wakeup load which is dependent on the synchronization table likewise is not relevant to Claim 1. And that's why we say, and Intel at least to that extent agrees with us, that the portion of the patent dealing with the synchronization of wakeup, the paired store pair load is not relevant to Claim 1. Thank you.

THE COURT: Thank you. Do you want to talk about in fact executed?

MR. HASLAM: Yes. I think if we go to column six of the patent and where we begin the detailed description of the invention, line 15, and this is something that both parties talked about, it's the simple three instructions. Instruction one is a store, instruction two is a load, instruction three is a load. And what the specification talks about here is that there are dependencies here, because we don't know what the contents of the registers R1, R2 and R3 are, those are the addresses which the store is going to load to or which the loads are going to retrieve from.

And if we go down to column 51, if we go down to

column 51, the patent says until you know the contents of those registers, the dependencies are ambiguous; that is, it cannot be determined whether there is in fact a dependency without knowing the contents of registers R1, R2 and R3, which cannot be adduced from the instructions alone. So right there I think you see something that we believe in this field to a person of ordinary skill in the art, it tells you what is the critical thing you need to do in order to be able to resolve the dependencies and that is the instructions have to execute far enough so that you know what memory address the load is going to retrieve from and the store has to execute far enough so that you know the memory address that the store is going to store to.

THE COURT: That's what I want to know. What are you doing when you're looking for the store? Are you -- Mr. Dally I think said you don't access the memory address, but you have to do something. How do you find out what's in there if you don't access it?

MR. HASLAM: If that's what he said --

MR. DALLY: Can I clarify?

MR. HASLAM: I can answer it or the expert can answer it.

THE COURT: Sure, why don't you.

MR. DALLY: So what I said is to detect the

conflict between the load and the store, you compare the two addresses. You compare the address of the load, in this case what would be in the register R2 or R3, to the address of the store. That is what is in register R1. So you're just taking the two addresses and seeing if they're the same. You don't have to actually go to the memory location to do that.

THE COURT: So how do you get the address?

MR. DALLY: The address is produced by the load/store execution unit, by reading the register. And in fact, it has to get the address first before it sends it to the memory system because the memory system needs the address to go and access the location.

THE COURT: So if I can go back to my moving van, I don't have to go to the house, I just have -- the first thing I have to do is find out if it's 142 Forest --

MR. DALLY: Exactly.

THE COURT: -- so I know --

MR. DALLY: The two moving vans, one of which is dropping a couch off, the other one of which is picking it up. You can simply look at your manifest and say the store, the dropoff, is to 142 Forest, and the load, the pickup is also 142 Forest. You detect the conflict back at the dispatch center. You don't

actually have to send the trucks out to do it.

THE COURT: Okay. Well that helps in one way. It still leaves me with the problem of deciding what in fact executed, what do you actually have to do to execute.

MR. HASLAM: In fact execute in the context of this claim with an out-of-order execution means that you have to have processed the instructions sufficiently far to determine the memory address that will be accessed.

THE COURT: So look in the phone book for the address.

MR. HASLAM: Look in the phone book. However, in the morning you come in and let's assume you're the load, you're the load to go out to pick up this sofa and they may actually send you out. I mean there's no dispute, both parties have indicated the instructions take different amounts of time to execute. So you send the first van out to 142, say go pick up what's there. Suddenly comes in a store after you sent the van out and it says oh, I've got to drop something off at 142. This load is not going to pick up the right stuff. So the system can then say at some point in time I'm going to want to either call him, stop him, when they get back tell them they've got to back or whatever. But you know there's a problem as soon as you know that the load is

going to pick something up at address 142 and that the store is going to drop something off there. And so you don't need to actually go to the place to pick it up in order to know.

THE COURT: I think I was following for awhile. I'm going to 142 expecting that the person who is dropping the sofa off has gotten there all right.

MR. HASLAM: Right, or that there will not be anything to be dropped off there.

THE COURT: So it's beyond looking up the address, I have to know what's going on at the address before I go out.

MR. HASLAM: You have to know what address you're going to.

THE COURT: Right.

MR. HASLAM: And then at some point you need to know one of two things to let that go ahead and do whatever it needs to do. Nobody is going to drop anything off at 142. In other words, nothing is going to happen there. There's no stores that are going to happen.

THE COURT: So going back to the computer now, the circuit, I know the address. Here's my little load. I know the address. How do I know that the address isn't going to work?

MR. HASLAM: Well, you know -- okay, you know the load is going to go and get some information at a particular location.

THE COURT: So I have to know more than the address. I start with knowing the address, but I have to know something more. Is it there yet?

MR. HASLAM: Well, but -- as a conclusion, yes, but the way it works is all you know when the load comes along is this load is going to go get some information at location one and it can start going out and trying to get that information. That's all you know at that point in time. Next along comes a store. Once you begin to execute the store, you execute the store far enough to say that this store is now going to store something at location one.

THE COURT: So that's where the execution takes place is at the store.

MR. HASLAM: Well, when you figured out that the store is going to store something at address one, you now know the addresses of the load and the store and you can see that they're both going to the same place. So you can determine that the load is going to get information, which is stale, because it won't have the store. It won't have the information that the store is going to put there for it. It's going to get the

information that was there before the store. So --

THE COURT: Okay. So at that point have I in fact executed?

MR. HASLAM: Yes. Once you have made a determination that the -- you know the addresses, you have -- you've executed as far as necessary to make a determination as to whether or not there's a data dependency or not as stated with respect to Table 1.

In the context of the language of column six, you know at that point in time what the contents of register one and register two are and you can now make a determination that if those two addresses, if R1 equals R2, they are both going to the same place. So in our view, you have executed, in the context of this art, to a person of ordinary skill in the art when you've gone far enough to know in the example of column six the contents of registers one, two and three which the patent says are the memory addresses that you're going to or that if you don't take it in the context of the registers is when you know the memory addresses that you're going to but you don't have to go there to know there's a problem.

THE COURT: Okay. So now I'm more confused than ever about your proposed construction. You're saying a load instruction is in fact executed before the

store instruction when the load instruction is actually accessed or is certain to access data that has not yet been updated by the store instruction?

MR. HASLAM: You can have a -- you can actually have the load having gone out and gotten the information. I guess the point that we're trying to capture is this: When you issue the load instruction, if we take the moving van, when you find out there's an order for 142 Forest, if that's what it was, but anyway 142 Forest, you send the van out there and the van starts driving out to 142 Forest, then the store comes along and you find out that the store is going to store something at 142 Forest, you don't -- at that point in time you know there's a conflict. But the van may have already gone to 142 Forest and come back, in which case you're going to say I'm sorry, that was a wasted trip.

THE COURT: So that's all that you mean. Intel has proposed a load construction is in fact executed when the load construction actually has loaded data from the memory location. That sounds to me like the whole thing worked. You got out there, the sofa was there, you picked it up.

MR. HASLAM: Right. And as we pointed out in our brief, the problem with that is because instructions take different lengths of time, let's suppose the van is

only five blocks away. Well, it's possible the van may have gotten there and be on its way back when you find out there's a conflict. Suppose he is driving 100 miles away. He may be 50 miles on his way there. The next thing comes in and says oh, we're going to store something at 142. Intel would say that that would not cause a data dependency or a conflict because that drive hasn't been completed, and as we showed in our brief, the problem with that is you would then let that load go out and come back because you wouldn't find the conflict under Intel because it hadn't completed, in their view of completed, and now you've got a problem because you think I've gone and picked up at 142 Forest, I don't have any problems, and you've missed the fact that there was a dropoff at 142.

And their argument that you would catch that somehow is wrong because once you've determined the storage address, you've decided what to do with the storage, it drops out of the circuit, and when the van finally gets back from its long trip out there, that store is no longer around anymore. So the fundamental difference is is we attempt to pick up the fact that the load instruction may be in the process of getting the information as well as may have already gotten the information when you can resolve the dependency by

figuring out what the load address is and the store address.

THE COURT: One of the things that seemed odd to me is that you talk about in fact executed, which is in the past tense, and yet you talk -- you wanted to include when it was certain to access data that has not yet been updated, which sounds like a conditional future-oriented sort of thing.

MR. HASLAM: As Professor Dally in his declaration stated a little bit today, in this field in fact executed has a meaning to people of skill in the art, and the process of executing an instruction has several aspects to it and can take a short period of time or a long period of time. It is in fact executed in the context of this particular art when it has begun execution and gone far enough to determine what the address is that it's going to go to, and that's why we said --

THE COURT: And then there's going to be a problem if there's nothing there, those two things.

MR. HASLAM: No, it will -- you will find the problem only when you know that it is going to the same place that a store is going to. It is possible, there's no dispute, it's possible that you're going to send -- you're going to execute a load and you're going to guess

that you're okay and there may be no problem, no store ever conflicted with it, in which case it goes out and it comes back. What we're talking about is that you determine in the context of Claim 1 that there's been a misspeculation at the point in time you know the address that the load went to or is going to is going to be the same as the address that the store is going to and the difference is Intel wants to say only after the load instruction has gone out and fetched the data and brought it back and we're saying that you can find that problem, you can learn what the two addresses are or the contents of register 1 and register 2 before you actually go out and fetch the data. You may have fetched the data and be sitting there waiting and determine there's a problem, in which case you've got to squash it, but you can also determine there's a problem without having to wait for the truck to go all way out there and come back and you can call them when they're 50 miles out there and same come back, you've got a problem. You don't have to let them go all the way out and come back, and that's the burden of our argument, and there are citations that are set forth in the brief.

But in column seven, there are several times where the patent talks about keeping track of operations as they are performed for purposes of the data speculation

circuit. Column seven, 24-29 talks about once the instructions have gone as far as possible, prior to reading from memory or requesting a store, the processing units notify the data speculation units so it can keep track of the operations. Column seven, line 45, it talks about having to make a determination that they have access to the same memory address, all of which we think supports the notion that a person of ordinary skill in the art would understand that in fact executed in the context of this means that you have executed the instructions sufficiently far that you can determine the addresses that they are going to go to or access.

THE COURT: Can I ask you just one thing which is not part of the construction, but you talk or someone talked about instruction windows.

MR. HASLAM: Yes.

THE COURT: Would you just tell me what that is?

MR. HASLAM: In the process of performing the instructions, part of the computer will go out and fetch instructions and bring them back and so the set of instructions that is there got ready to operate on what's called the instruction window. So it may be, I'm in quicksand, but hypothetically you could have ten

instructions you're dealing with, maybe 100 instructions, I don't know what the size might be, but those are the ones you've been operating on. You're looking for any misspeculations or in that particular one, the instructions in that particular window.

THE COURT: And then, because we're really running out of time, when we talk about predictor, I wondered whether your content is is it an issue in this case that defendant's accused device produces a prediction not based on historical misspeculation? Because if it doesn't, I don't see why we're worrying about this term.

MR. HASLAM: I don't know why we need to construe predictor or prediction. I think the claim is pretty clear. I mean the predictor is a circuit that receives a misspeculation. The prediction is associated with the data consuming instruction and based on misspeculation indication. I mean we both are arguing over something that, in our view if you've made the determination about how and whether you're going to put load/store pairs in the claim and once you decide what in fact executed is, we believe you don't need to construe the rest of it and all we've done is --z the parties have done is sort of recast the claim is pretty clear in a different language.

THE COURT: If I construed prediction as a variable that indicates the likelihood that the data speculative execution of the load instruction will result in a misspeculation, would that make sense?

MR. HASLAM: Yes.

THE COURT: All right. Thank you. Mr. Lee, you'll be finishing up?

MR. LEE: Yes. I think I can do it in six minutes, Your Honor.

MR. HASLAM: Sorry, Bill. If I would have known that, I would have talked a few minutes longer.

MR. LEE: He's an old friend and now he has wasted 30 seconds of it. Your Honor, four points are that in fact executed, if I can take Your Honor to Slide 7 of our presentation.

First, the words are in fact executed, and Your Honor will see that in the claim, different tenses and different voices are used by the patentee. One of the things the federal circuit has been relatively clear about is the patentee has the power of the pen and the purpose of the claim is to give public notice, so we ought to read the words in the manner in which they're drafted and the words are in fact executed are in the past tense.

Compare if you would, Your Honor, paragraphs A and

B which are stated in the present tense as if something is happening at the same time. That's why in fact executed is something that has occurred.

The second point is the idea that Professor Dally and Mr. Haslam just advocated which is if you know the address is 142, that's not even consistent with what their claim interpretation is. Their claim interpretation is either you've got the 142 Forest or the truck is on its way and it's certain to arrive there. And the difference between the two is instructive for two reasons. One, I think it belies the argument that in fact executed has this clarion meaning to one of skill in the art because it has been described three different ways.

The second is that it tells Your Honor that they're trying to do something with the past tense in fact executed that the term won't support. Indeed, Your Honor, if what they wanted to say is all you need is the address, they could have said all you need is the address. They said not just executed, they said in fact executed.

Point number three, Your Honor. The specification, if I could turn your Honor to -- I'll bring us to Slide 10 -- the specification tells us what it means to actually execute a load instruction and it loads the

content of memory location A(1). We're just using in our claim term what they say it means to execute the load and using it in precisely --

THE COURT: But something doesn't make sense because the claim itself talks about is in fact executed before the data producing instruction.

MR. LEE: So that means that it's gone to the register. To use our moving van example, your Honor, it could be that if you're going to pick up -- if I sent the truck to pick up my daughter's furniture to take it to graduate school, it could be that the truck arrives there and her furniture is not yet there. Right? But it could be that my other daughter's furniture is there and what they do is pick that up instead. That's Mr. Steinberg's first example that if the store instruction is dependent -- the load is dependent upon the store, you can go to the store and you can get the wrong information. That's why this in fact executed has some meaning, and that's why when they want to talk in the past tense, they do. They did at Slide 10 and they also do at Slide 11. And it's our interpretation of in fact executed is giving meaning to the past tense, it's giving meaning to in fact, and it really is taking words from the specification from the only example to say okay, we'll take you at your word, here is what you

described, and we'll give it that meaning.

I think, Your Honor, the fourth point is the best demonstration that we're actually sort of two ships passing in the night is to compare the manner in which the parties are articulating the terms in their brief. If I could have Slide 12. On Slide 12, Your Honor, we have excerpted the portions of WARF's brief where they have tried to define in fact executed for Your Honor and it's bound to access, talking about something in the future; started the process, suggesting that something is begun but not completed; being performed, present tense; at least partly executed, which is just the antithesis or one of the antitheses of start executed; starts execution, present tense, progressed to a point.

So if Your Honor takes the language as it is, recognizes that they have the power of 10, takes it on its face as past tense and then look at what they describe as loading, our claim interpretation is consistent with the intrinsic evidence in the specification.

THE COURT: But there is this problem because there's going to be misspeculation data for the prediction table in situations other than when you get there and it's old information or the wrong information. If there's nothing there and you can't ever load

anything, that's a misspeculation.

MR. LEE: There will, but I think that there will always be something there, even if it's a zero. So even if you load a zero, in our example of the bank account, if they haven't loaded my -- if the request is what happened the month of July and they haven't loaded my paycheck for the month of July and there's a zero in there because there have been no deposits from July, what it will do is access the zero and you'll have a misspeculation.

Your Honor, to get myself done on time, if you in fact executed, if you -- if Your Honor goes back to Mr. Steinberg's tutorial and the manner in which -- let me step back. If you take the summary of the invention which, while many of them as Mr. Haslam says has a little bit for everybody, if you take this summary of the invention and read it against 4, 5 and 7, it is very consistent. It's consistent in purpose, it's consistent in terms, it's consistent in pairs, and it's consistent about what has happened to the construction, and I think Your Honor will find it's consistent with our claim interpretation.

To go to the very last issue Your Honor had raised, we had put together Slides 45 and 46 which goes to this issue of the predictor, and let me just say this, Your

Honor: We do think that the predictor needs to be based on a past history of misspeculations for two reasons. That's what the patent says -- three reasons. That's what the patents says, it's the only disclosed embodiment, and more critically that's the way the invention works. Thank you, Your Honor.

THE COURT: Could I just ask you, Mr. Lee --

MR. LEE: Yes.

THE COURT: -- is it your understanding the predictor is used at times -- if the -- if there is no misspeculation, that changes the predictor as well as when there is a misspeculation?

MR. LEE: Yes. If there is no misspeculation, then the predictor will probably go down.

THE COURT: Right.

MR. LEE: Or if there's misspeculation based upon a different pair, it will go down.

THE COURT: Okay.

MR. LEE: Thank you, Your Honor.

MR. HASLAM: Thank you.

THE COURT: Thank you all very much. Did you want to say one more thing, Mr. Haslam?

MR. HASLAM: Just -- I never turn down that offer.

MR. LEE: Now we're going to count -- we're

going to see if we agree with what one means.

THE COURT: It's very rare there's just one thing.

MR. HASLAM: I don't know if Mr. Lee took us to task for the language used in the claim like some of his arguments. On his last argument about the history of misspeculations, we were clear in the claim as to what we meant. We didn't put history in there and now he clearly wants to take history and speculation from the specification and put it into the patent claim when it's not there. That's why we think you don't need to construe prediction or predictions.

THE COURT: Thank you. This has been very helpful and I'll take it under advisement and get something out relatively soon. No promises.

(Proceedings ended at 12:02 p.m.)

I, LYNETTE SWENSON, Certified Realtime and Merit Reporter in and for the State of Wisconsin, certify that the foregoing is a true and accurate record of the proceedings held on the 8th day of August, 2008, before the Honorable Barbara B. Crabb, Chief Judge of the Western District of Wisconsin, in my presence and reduced to writing in accordance with my stenographic notes made at said time and place.

Dated this 12th day of August 2008.

____

Lynette Swenson, CRR, RMR,
RPR, CBC
Federal Court Reporter

The foregoing certification of this transcript does not apply to any reproduction of the same by any means unless under the direct control and/or direction of the certifying reporter.